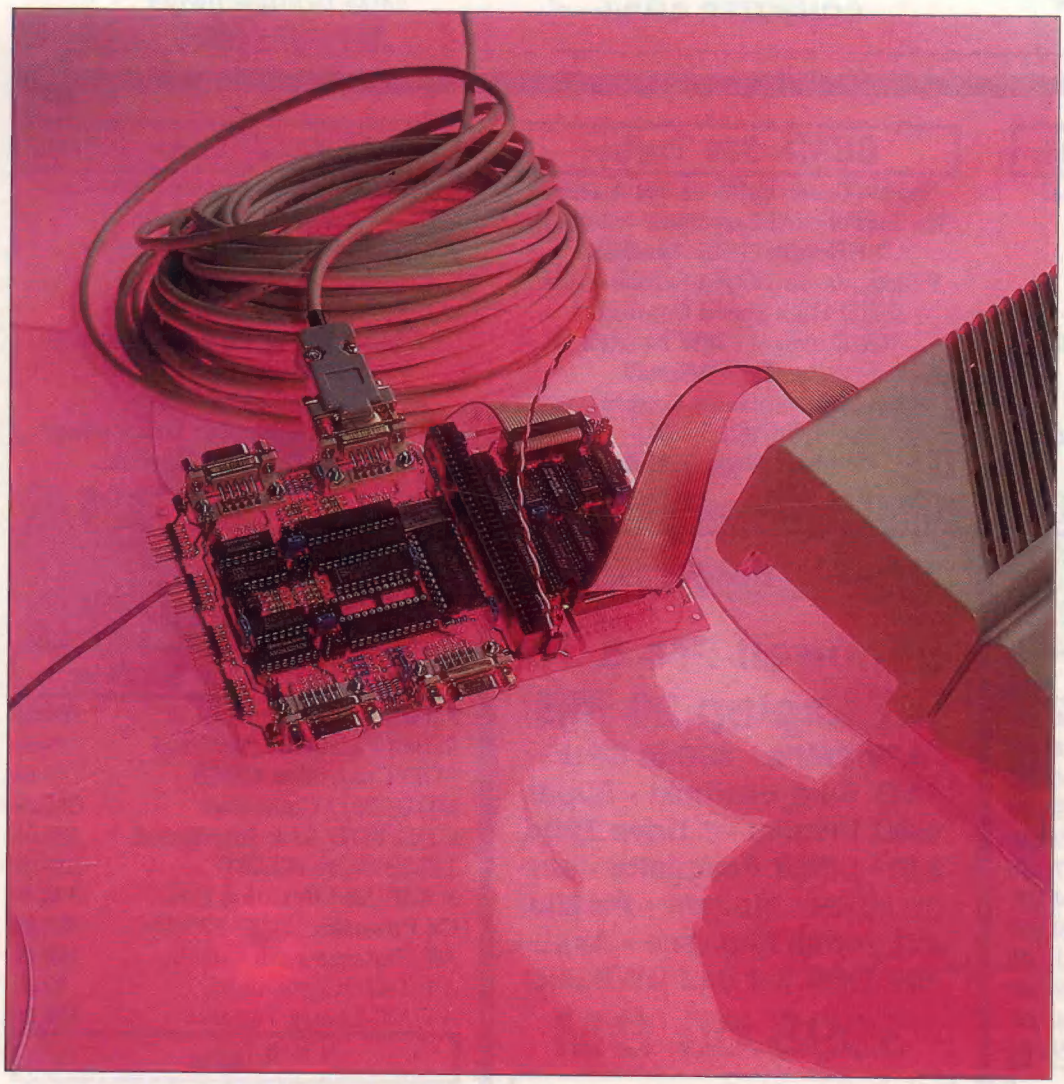


EingekLinkt

Teil 1: Link-Adapter mit 'OMTI-Slot' für Atari ST

Helge Schulz

Die Links, schnelle serielle Schnittstellen, sind das Tor zur Welt der Transputer. Gewöhnliche Computer brauchen geeignete Adapter, um dorthin vorzudringen. Unser Atari-Link-Adapter (ALIA) besitzt gleich vier Link-Anschlüsse und kann deshalb nicht nur als Verbindung zu einem Transputersystem genutzt werden, sondern auch zur Vernetzung von Atari STs untereinander oder mit anderen 'gelinkten' Geräten. Im DMA-Modus werden bis zu 1,6 MByte/s übertragen. Der DMA-Port ist durchgeschleift und gepuffert, so daß weitere DMA-Geräte (Festplatte, Laserdrucker) angeschlossen werden können. Dieser Schaltungsteil, der auch ohne Links verwendbar ist, bietet außerdem einen Slot zum Anschluß eines billigen PC-Festplattencontrollers.

Die Hardware des Link-Adapters steht in diesem ersten Teil unserer Beitragsreihe im Vordergrund. In den nächsten Heften geht es dann um die Software für verschiedene Anwendungen – als da wären: Anschluß einer Transputerkarte TEK 4/8 oder KEK 4/8 und Einsatz des Occam-Programmiersystems OCS, Betrieb eines OMTI-Harddisk-Controllers und Vernetzung von STs untereinander.

Der DMA-Port

Der DMA-Port des Atari ST hat bisher in der Beliebtheitsskala der Hardware-Entwickler einen der letzten Plätze belegt. Ein Grund dafür ist sicherlich, daß vom Hersteller kaum Informationen über diese Schnittstelle veröffentlicht wurden. Ja, selbst im Handbuch ist nur von einem Harddisk-Anschluß die Rede. Dabei handelt es sich bei dem ACSI [1] um eine vielseitig nutzbare, echte bidirektionale 8-Bit-Schnittstelle.

Rechnerintern wird der Port vom DMA-Controller bedient, der zu den vier herstellerspezifischen Bausteinen im Atari ST zählt. Er steuert die gesamte Datenübertragung zwischen Prozessor, Speicher und den am DMA-Port angeschlossenen Geräten. Auch sorgt er für den Datenaustausch mit dem Floppy-Controller; bei der Programmierung des DMA-Controllers muß man dies beachten, um den Diskettenbetrieb nicht zu stören. Leider hat der Baustein auch seine Schwachstellen und Macken; insgesamt macht dieser Teil des Atari ST einen eher unausgereiften Eindruck. Dies schlägt sich auch im Betriebssystem nieder, wo nur eine Routine vorhanden ist, die den Festplatten-Bootsektor laden und starten kann. Hat Atari vielleicht deshalb mit Informationen gegeizt, weil hier mit einer zu heißen Nadel gestrickt wurde?

Das Informationsdefizit hatte Konsequenzen. Zum einen gibt es bis jetzt nur recht wenige Geräte für den DMA-Port, zum anderen gestatten die meisten davon nicht den gleichzeitigen Betrieb von mehreren Einheiten. Der c't-OMTI-Adapter [2] ist ein Beispiel dafür. Aber selbst wenn man weiß, wie die Geräte am DMA-Port zu adressieren sind, hält das ACSI-Konzept noch manche Tücke bereit. Beispielsweise genügt es nicht, alle Anschlüsse einfach durchzuschleifen; das funktionierte zwar mit der alten Atari-Festplatte SH204, aber nicht mehr mit der neuen SH205. Warum dies so ist, zeigt sich bei näherer Betrachtung des Funktionsprinzips.

Lauschaktion

Jedes Gerät am DMA-Port muß ständig die Schnittstelle daraufhin abhorchen, ob vom Rechner ein Kommando gesendet wird. Damit dies im Datenstrom erkennbar wird, ist eine Adreßleitung mit der Bezeichnung A1 herausgeführt, die auf logisch 0 geht, wenn das erste Byte eines

Kommandos auf den Datenleitungen liegt. Sonst bleibt A1 ständig auf logisch 1. In den Bits 7, 6 und 5 des ersten Kommando-Bytes wird die Nummer für einen von maximal acht Controllern übertragen.

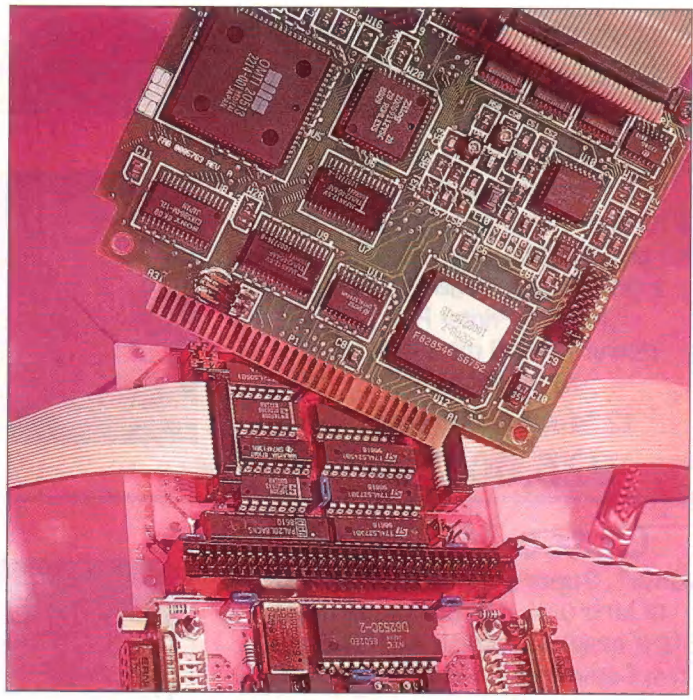
Ein Gerät am DMA-Port muß also, wenn noch andere angeschlossen sind, auf jeden Fall das erste Byte eines jeden Kommandos mitlesen. Dann muß es die höchstwertigen drei Bits dieses Bytes mit der eigenen Controller-Nummer vergleichen. Stimmt die Nummer überein, wird das entsprechende Gerät aktiv und wickelt den Datenaustausch mit dem Rechner ab. Andernfalls darf es auf keinen Fall irgendwelche Signale auf den DMA-Port legen, um den Datenverkehr mit dem aktiven Gerät nicht zu stören.

Crash-Gefahr

Hängt nun eine Kette von Geräten am DMA-Port, so kann ein inaktives Gerät nicht erkennen, ob ein Gerät vor oder hinter ihm aktiv ist. Handelt es sich dabei um die Atari-Festplatte SH205, so ist ein 'Daten-Crash' fast unvermeidlich: bei dieser werden nämlich die Datenleitungen zwischen Ein- und Ausgang gepuffert. Befindet sich vor der SH205 in der Kette ein zur Zeit aktives Gerät, das den DMA-Port nur einfach durchschleift und gerade Daten zum Rechner senden will, kommt es unweigerlich zu einem Konflikt auf den Datenleitungen; das aktive Gerät legt Daten auf den Bus, gleichzeitig sind aber die Treiber der SH205 in Richtung Rechner geschaltet.

Ein weiteres Problem liegt darin, daß es am DMA-Port kein Signal gibt, welches anzeigt, daß eventuelle Treiber in den Datenleitungen eingeschaltet werden können. Besonders problematisch wird es, wenn Daten in den Rechner übertragen werden sollen. Die Treiber dürfen erst eingeschaltet werden, wenn das Signal NSel (Select) aktiv wird, um nicht eine Datenkollision mit dem Disk-Controller zu riskieren. Die Zeit zum Bereitstellen der Daten ist aber äußerst knapp (rund 200 ns). Darum sollte man die Signale D0 . . . D8, NotSel und NotAck nicht unnötig verzögern, damit auch noch die Daten des letzten Gerätes in der Kette korrekt gelesen werden können.

Auf irgendeine Weise muß man aber in die Signalleitungen zwischen dem DMA-Eingang und -Ausgang eingreifen, wenn auch der Betrieb mit einer SH205 möglich sein soll. Um diese Nuß zu knacken, haben wir uns zunächst die Schaltung der Atari-Festplatte genauer angesehen. Bei dieser werden die Datenleitungen kurz vor dem DMA-Ausgang durch einen 74LS245-Treiber gepuffert. Wird die Festplatte mit der eingestellten Controller-Nummer



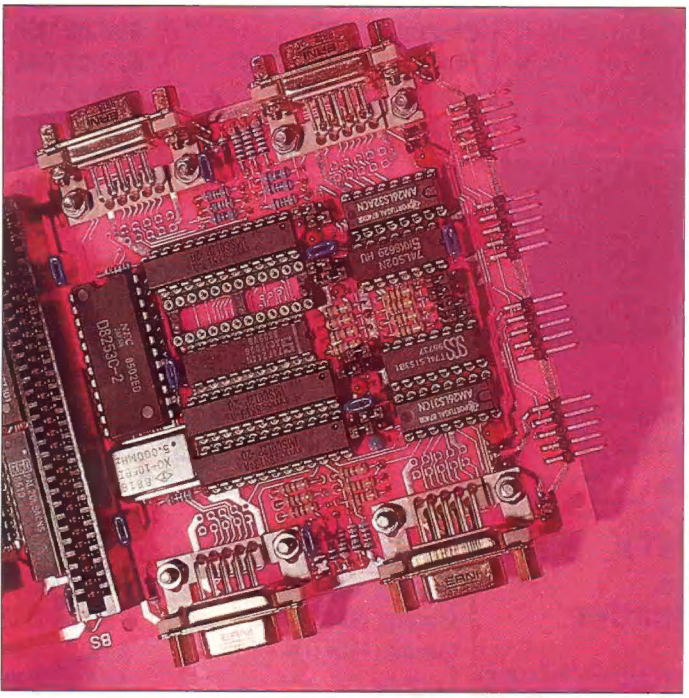
aktiviert, so wird dieser Treiber einfach abgeschaltet. Das bedeutet aber, daß der SH205-Controller ständig ungepuffert vor dem Treiber angeschlossen ist! Die übrigen Leitungen sind durchgeführt. Zwar hat man so am DMA-Ausgang der Festplatte gepufferte Datenleitungen, aber der DMA-Controller im Rechner wird unnötig belastet, weil in der SH205 vier TTL-ICs angeschlossen sind. Natürlich könnte man zur Pufferung am Eingang noch einen Treiber zwischenschalten. Dadurch werden aber die Signale auf den Datenleitungen verzögert, und die Timing-Bedingungen für den Betrieb mehrerer Geräte am DMA-Port würden erheblich verschlechtert.

Das DMA-Port-Interface mit einstellbarer Controller-Adresse und Pufferung aller kritischen Signale belegt ein gutes Drittel der Karte. Wer nur einen OMTI-Controller betreiben möchte, braucht lediglich diesen Teil.

Ausgetrickst

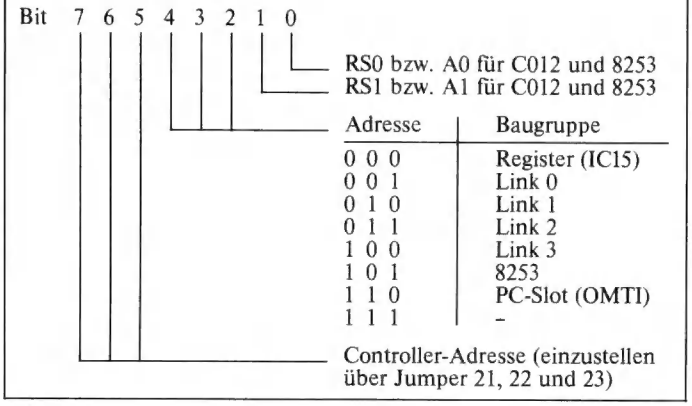
Beim c't-ALIA werden die Datenleitungen am Eingang gepuffert. So wird dem treiberschwachen DMA-Controller nur eine TTL-Last zugemutet. Um das beschriebene Problem mit der SH205 zu meistern, manipulieren wir das Schreib-Lese-Signal (RNotW) für die nachfolgenden Geräte. Zuvor wird dieses Signal aber selbst durch einen Treiber vom Typ 74F244 (IC18) gepuffert. Dasselbe geschieht mit NotRes, A1, NotSel und NotAck. Der 74F244 bietet neben seiner Schnelligkeit (4 ns Durchlauf-Verzögerung) den Vorzug, Eingänge mit Schmitt-Trigger-Charakteristik zu besitzen. Alle zeitkritischen Signale sind außerdem zum Schutz gegen Reflektionen über Dämpfungswiderstände (R33 . . . 35) geführt.

Das Signal RNotW gelangt hinter dem Treiber auf ein UND-Glied vom Typ 74F08, dessen Ausgang dann das RNotW des DMA-Ausgangs darstellt. Das UND-Glied sorgt dafür, daß ein



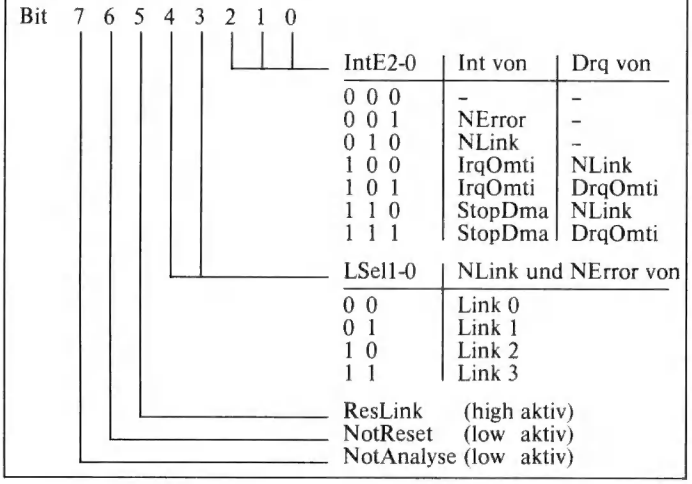
Zwei Links, zwei rechts: bis zu vier 'Aalener Links' passen auf die Karte. Für kurze Verbindungen reichen aber auch die einfachen Link-Anschlüsse an der Stirnseite.

Status-Byte (IC14)



Adresse	Baugruppe
0 0 0	Register (IC15)
0 0 1	Link 0
0 1 0	Link 1
0 1 1	Link 2
1 0 0	Link 3
1 0 1	8253
1 1 0	PC-Slot (OMTI)
1 1 1	-

Register-Byte (IC15)



IntE2-0	Int von	Drq von
0 0 0	-	-
0 0 1	NError	-
0 1 0	NLink	-
1 0 0	IrqOmti	NLink
1 0 1	IrqOmti	DrqOmti
1 1 0	StopDma	NLink
1 1 1	StopDma	DrqOmti

LSel1-0	NLink und NError von
0 0	Link 0
0 1	Link 1
1 0	Link 2
1 1	Link 3

Diese Funktionen haben die einzelnen Bits in den beiden Latches.

Lese-Signal (RNotW=0) auf jeden Fall weitergegeben wird. Im Normalfall wird der zweite Eingang des UND-Gatters (RWE) vom PAL 20L8 (IC11) auf logisch 1 gehalten; dann wird das RNotW-Signal einfach weitergegeben. Ist der Link-Adapter aber aktiv und sollen Daten zum Atari übertragen werden, so geht RWE auf 0 und somit auch RNotW für die nachfolgenden Geräte. Damit wird diesen Geräten ein Schreibzugriff vorgetäuscht; ein Kurzschluß auf den Datenleitungen wird so verhindert.

Den nachfolgenden Geräten macht dieser Schreibzugriff nichts aus, weil sie ja nicht aktiviert sind. Das zweite eingesetzte UND-Glied steuert den Eingang NotE des Datenpuffers 74LS245 (IC12). Es sorgt durch die Verknüpfung mit RNotW und dem Signal DataE des PALs dafür, daß zu schreibende Daten sofort weitergereicht werden.

In Richtung Atari darf der 74LS245 nur treiben, wenn NotSel bei einem Einzelbytezugriff oder NotAck beim DMA-Zugriff aktiv wird und gleichzeitig A1=1 ist. Dafür sorgt das PAL mit dem Signal DataE. Sicherlich könnte man auch die logische Funktion der beiden UND-Glieder noch ins PAL hineinpacken. Darauf haben wir aber verzichtet, um kleinstmögliche Durchlauf-Verzögerungen zu erreichen.

Schneefrei

Die Reset-Leitung NotRes des DMA-Ports ist direkt mit der Reset-Leitung der 68000 verbunden. Diese Leitung ist bidirektional ausgelegt: wenn die CPU einen Reset-Befehl bekommt, zieht sie diese Leitung auf 0; es kann aber auch durch einen 0-Impuls von außen ein Reset ausgelöst werden. Es wurde keine Pufferung zum DMA-Ausgang hin vorgesehen, um die Möglichkeit, von einem DMA-Gerät einen zentralen Reset auslösen zu können, nicht zu verbauen.

An derselben Reset-Leitung hängt aber auch der Video-Shifter, der beim ST für den Bildschirmaufbau sorgt. Dieser Atari-spezifische Chip scheint bei einigen Geräten sehr empfindlich auf das Reset-Signal zu reagieren. Das äußert sich etwa darin, daß schon das Übersprechen auf einem etwa 1 m langen Kabel (beispielsweise von einer SH205) mit offenem Ende dazu führen kann, daß man nur noch Schnee auf dem Bildschirm sieht. Mit dem LC-Filter aus L1 und C5 sollen solche unangenehmen Störungen auf dem Link-Adapter abgefangen werden. Tritt der Störeffekt trotzdem auf, muß man für L1 und C5 Bauteile mit höherer Induktivität beziehungsweise Kapazität vorsehen; notfalls muß man die vorhandenen Kabel kürzen. Das Kabel zwischen DMA-Port und ALIA sollte grundsätzlich so kurz wie möglich gehalten werden (nicht mehr als 1 m insgesamt), sonst riskiert man ähnliche Effekte auch zwischen den übrigen Leitungen, was zu Übertragungsfehlern führen kann.

Ausgewählt

Die Controller-Nummer wird auf dem Link-Adapter durch die Jumper 21, 22 und 23 eingestellt. Das erste Byte einer Befehlsfolge, das mit A1=0 vom Atari gesendet wird, bleibt in einem Latch vom Typ 74LS273 (IC14) gespeichert. Dieses Byte enthält in den drei höchstwertigen Bits die Nummer des ange-

Register des C012

RS1	RS0	Bedeutung
0	0	InputDataRegister
0	1	OutputDataRegister
1	0	Bit 7 6 5 4 3 2 1 0 — DataPresent (Input), InputInterruptEnable
1	1	Bit 7 6 5 4 3 2 1 0 — Output Ready, OutputInterruptEnable

Register des 8253

A1	A0	Bedeutung
0	0	DataCounter 0
0	1	DataCounter 1
1	0	DataCounter 2
1	1	Bit 7 6 5 4 3 2 1 0

Bit 0: 0 Binär, 1 Dezimal
Bit 1 2 3: Mode (hier 0 0 0)
Bit 4 5: 0 0 Counter Stop, 0 1 Low Byte, 1 0 High Byte, 1 1 Low/High Byte
Bit 6 7: Status für 0 0 Counter0, 0 1 Counter1, 1 0 Counter2, 1 1 -

Die Register des Link-Bausteins und des Zählers in Kurzform

sprochenen Controllers. Sie wird mit Hilfe der drei Inverter von IC13 und zwei UND-Gliedern von IC17 mit der eingestellten verglichen. Stimmen die Nummern überein, so geht das Signal Akt des PALs auf logisch 1. Die LED leuchtet auf und zeigt damit an, daß der Adapter aktiviert ist.

Die Bits 2 bis 4 des in IC14 gespeicherten Bytes dienen zur Adressierung der verschiedenen Baugruppen auf dem Adapter. Sie werden dem 3-zu-8-Decoder 74LS138 (IC16) zugeführt. Die Bits 0 und 1 adressieren die Register des Zählers 8253 oder des Link-Bausteins C012 über A0, A1 beziehungsweise RS0, RS1.

Der Freigabeeingang des Decoders ist mit dem PAL-Ausgang NDmaSel verbunden. Geht NDmaSel auf 0, so erfolgt ein Zugriff auf den Link-Adapter. Dieses 'Chip Select' wird dann über den Decoder an die adressierte Baugruppe weitergegeben. Die erste Adresse innerhalb des Link-Adapters ist mit einem zweiten Latch (IC15) belegt, dessen Ausgänge verschiedene Funktionen haben (siehe Kasten).

Beim DMA-Interface kommt es auf optimales Zeitverhalten an; die kritischen Signale sind deshalb über Treiber vom F-Typ geführt.

Abgezählt

Experten werden sich beim ersten Blick auf den Schaltplan schon gewundert haben, was ein Zählerbaustein vom Typ 8253 (IC5) auf dem Link-Adapter zu suchen haben mag. – Mit diesem Zähler wird eine Schwäche des DMA-Controllers ausgebügelt: dieser kann nämlich im DMA-Modus nur in Blöcken von 512 Bytes empfangen oder senden (und davon höchstens 255, also 130 560 Bytes); außerdem funktioniert das Abzählen nicht richtig. Sendet der DMA-Controller beispielsweise einen Block im DMA-Modus an ein Gerät und fordert das Gerät danach über Drq (Data Request) weitere Daten an, so liefert der Controller auch noch weitere Bytes mit dem Wert 0. Das heißt, der Controller beendet den DMA-Transfer nicht korrekt. Zwar unterbleibt der Zugriff auf den Speicher, was bei der Übertragung in den Rechner sehr wichtig ist, weil sonst womöglich

Link-Konfiguration

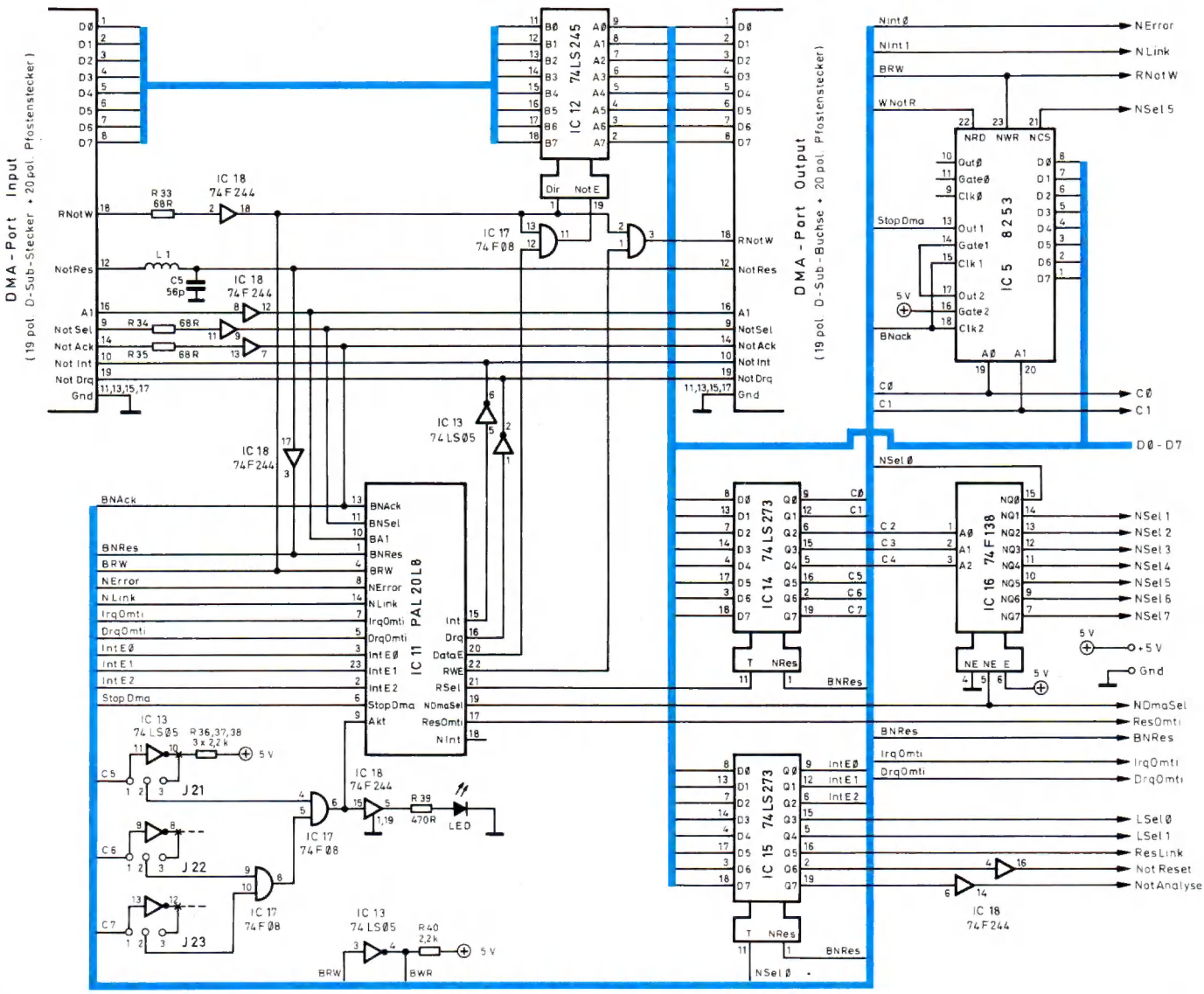
Bedeutung	Link-Nr. 0	1	2	3
+5V-Anschluß (1-2 verbunden)				
einf. Link	J1	J6	J11	J16
Aalener Link	J2	J7	J12	J17
Aalener Link (1-2 verbunden)				
einf. Link (2-3 verbunden)				
LinkInput	J3	J8	J13	J18
NotError	J5	J10	J15	J20
Übertragungsrate (10 MB: 1-2, 20 MB: 2-3)	J4	J9	J14	J19

Controller-Nummer („1": 1-2, „0": 2-3)

J21	J22	J23	Nummer
0	0	0	0
0	0	1	1
0	1	0	2
	·		·
	·		·
	·		·
1	1	1	7

„OMTI-Slot"

J24	AEN an Q0 von IC14
J25	AEN an Masse



Stückliste

Halbleiter, ICs

IC1 ... IC4	IMS C012-P20S
IC5	8253 (2 MHz)
IC6	74 LS 153
IC7	26 LS 31
IC8	26 LS 32
IC9, IC18	74 F 244
IC10	74 LS 02
IC11	PAL 20L8-A
IC12	74 LS 245
IC13	74 LS 05
IC14, IC15	74 LS 273
IC16	74 F 138
IC17	74 F 08
IC19	74 F 32
X1	Quarz-Oszillator 5 MHz
LED	gelb

Widerstände

R1, R3, R6, R9, R11, R14, R17, R19, R22, R25, R27, R30	1 kOhm
R2, R4, R5, R7, R10, R12, R13, R15, R18, R20, R21, R23, R26, R28, R29, R31	100 Ohm
R8, R16, R24, R32	680 Ohm
R33 ... R35	68 Ohm
R36 ... R38	2,2 kOhm
R39	470 Ohm
R40	2,2 kOhm

Kondensatoren

14 x CB	100 nF
1 x CB	10 µF (Tantal)
C1 ... C4	1 µF (Tantal)
C5	56 pF (Keramik)

Diverses

L1	Miniaturdrossel 68 mH

brechbare Pfostenstecker-leisten für Jumper
24 Steckbrücken
IC-Fassungen (DIL)
4 x 14polig
4 x 16polig
5 x 20polig
1 x 24polig
5 x 24polig (schmal)
62poliger PC-Slot
19poliger Sub-D-Stecker, 19poliger Sub-D-Buchse (90°, Printmontage) oder 19poliger Sub-D-Stecker, 19polige Sub-D-Buchse (gerade)
20poliges Flachbandkabel
2 x 20poliger Platinenverbinder für Flachbandkabel
4 x 10poliger Pfostenstecker (90° oder gerade)
4 x 9polige Sub-D-Buchse (90°, Printmontage) oder
4 x 9polige Sub-D-Buchse (quetschbar)
9poliges Flachbandkabel
4 x 10poliger Platinenverbinder für Flachbandkabel

wichtige Daten überschrieben würden, nach außen hin läuft die DMA-Übertragung aber voll weiter.

Normalerweise macht dieses Verhalten keine Probleme, weil der angeschlossene Harddisk- oder Disk-Controller die Übertragung von sich aus stoppt. Bei den Links gibt es aber Probleme. Soll ein Link zum Beispiel zum Anschluß einer Transputerkarte dienen, so müssen die Bytes schon genau abgezählt werden, damit ein vorhandenes Protokoll auch bei DMA-Transfer zu realisieren ist; sonst wäre die DMA-Fähigkeit hierfür nutzlos.

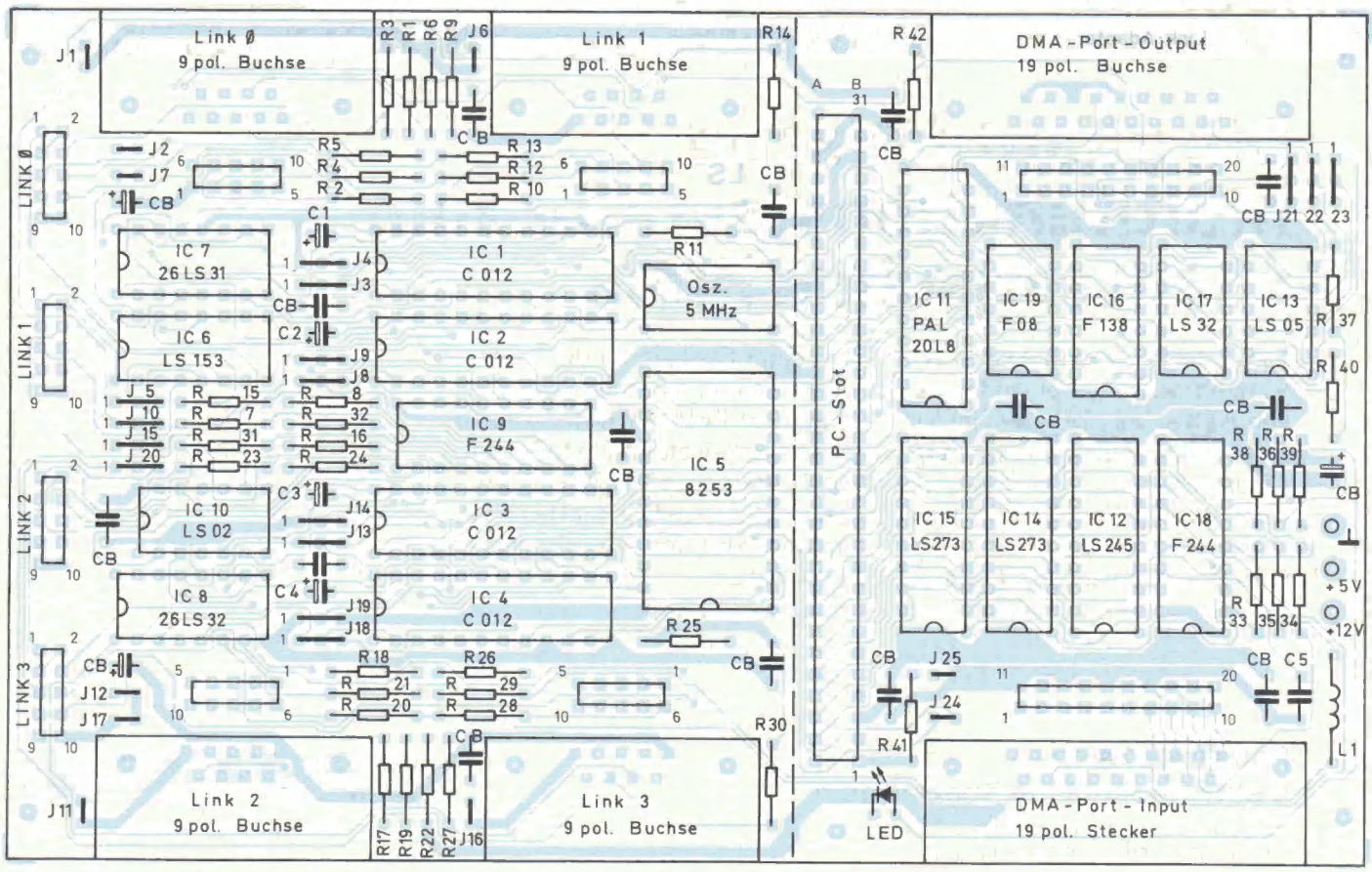
Der Zähler 8253 gleicht dieses Manko aus. Er zählt die übertragenen Bytes und blockiert das Drq-Signal im richtigen Moment über den Ausgang StopDma. Als Zählsignal dient NotAck (NotAcknowledge), das 'Chip Select' für den DMA-Betrieb, das mit den Eingängen von Zähler 1 und 2 des 8253 verbunden ist.

Der 8253 besitzt drei unabhängige 16-Bit-Binärzähler. Sie sind eigentlich nur als Timer ausgelegt, also zum Betrieb mit einem regelmäßigen Takt, was sich dadurch bemerkbar macht, daß der eingeschriebene Zählwert erst mit dem ersten Eingangssignal in den eigentlichen Zähler übernommen wird. Dies muß man natürlich bei der Programmierung, aber auch bei der Beschaltung, beachten: direktes Kaskadieren ist unmöglich. In der ALIA-Schaltung ist deshalb der Freigabeeingang (Gate) des Zählers 1 mit dem Ausgang OUT des Zählers 2 verbunden. So addieren sich die Zählwerte, denn Zähler 1 beginnt erst zu laufen, wenn Zähler 2 bei Null angekommen ist. Erreicht Zähler 1 schließlich auch die Null, geht StopDma auf 1.

Links, Links ...

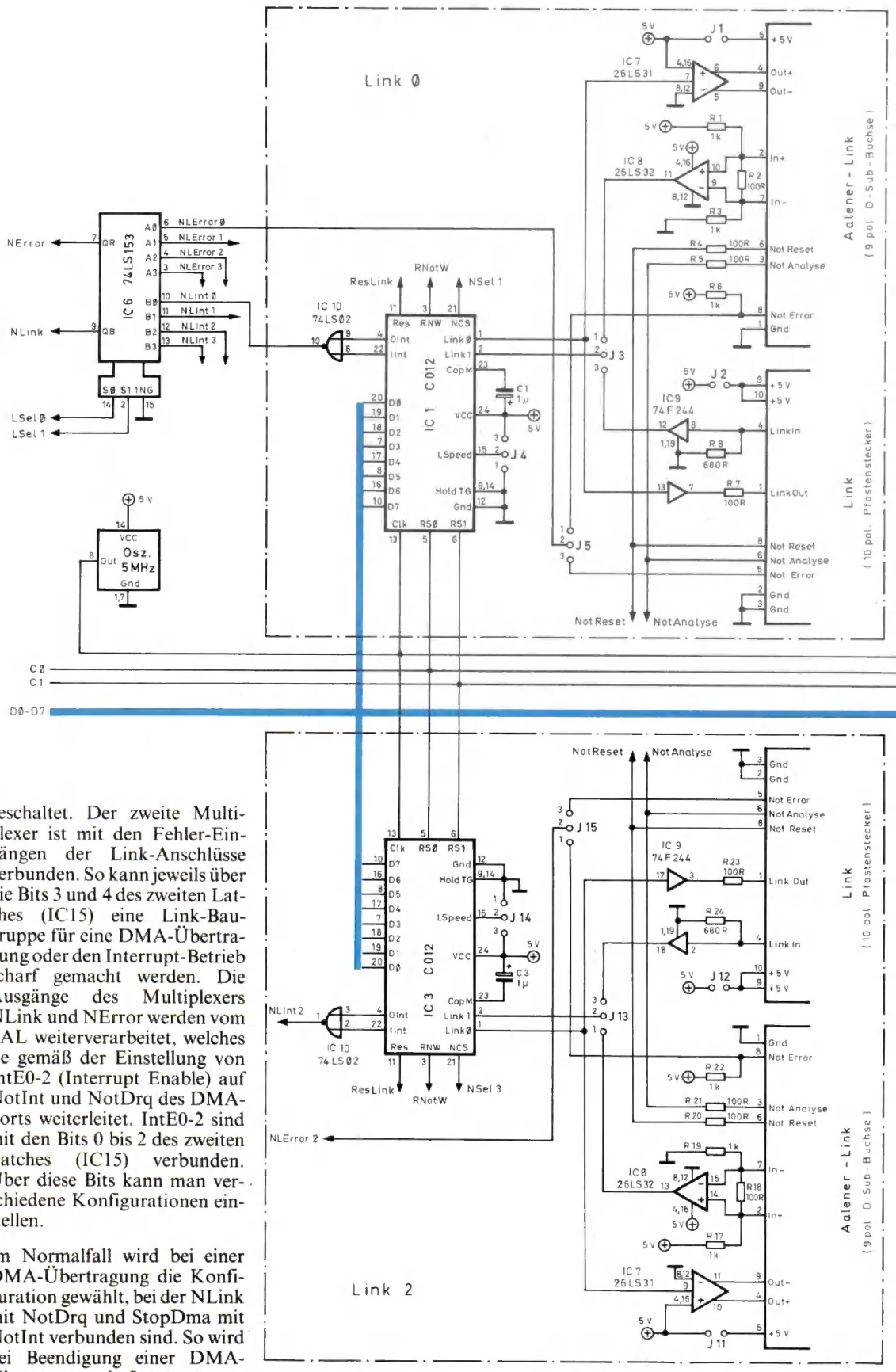
Die Schaltung ist für vier Links ausgelegt, so daß bei Maximal-

Nur die Bauelemente rechts von der Trennlinie, mit Ausnahme der Slot-Buchse, zählen zur Pflichtbestückung, alles übrige gehört zur 'Kür' – kann also dem Bedarf angepaßt werden.



ausbau dieselben Konfigurationsmöglichkeiten bestehen wie bei einem Transputer-Baustein. Sie kann aber je nach Bedarf auch teilbestückt betrieben werden. Jeder Link-Baustein C012 [3] ist mit zwei Links- sich um den üblichen 10poligen Pfostenstecker, wie er auch auf per auszuwählen ist. Bei dem einen Link-Anschluß handelt es sich um dem üblichen 10poligen Pfostenstecker, wie er auch auf der TEK 4/8 zu finden ist. Der andere Anschluß ist ein 'Aalener Link-Anschluß'[5]: er entspricht dem von der DOIT (Deutsche Occam-Interessengemeinschaft der Transputeranwender) Ende 1987 in Aalen definierten Standard. Auf der ALIA-Karte wurden davon vier sogenannte Master-Buchsen realisiert. Das Besondere am Aalener Link ist, daß zur Übertragung der Daten Differential-Treiber und -Empfänger (RS-422) eingesetzt werden. Damit können sehr viel längere Kabel benutzt werden als beim 'normalen' Link-Anschluß.

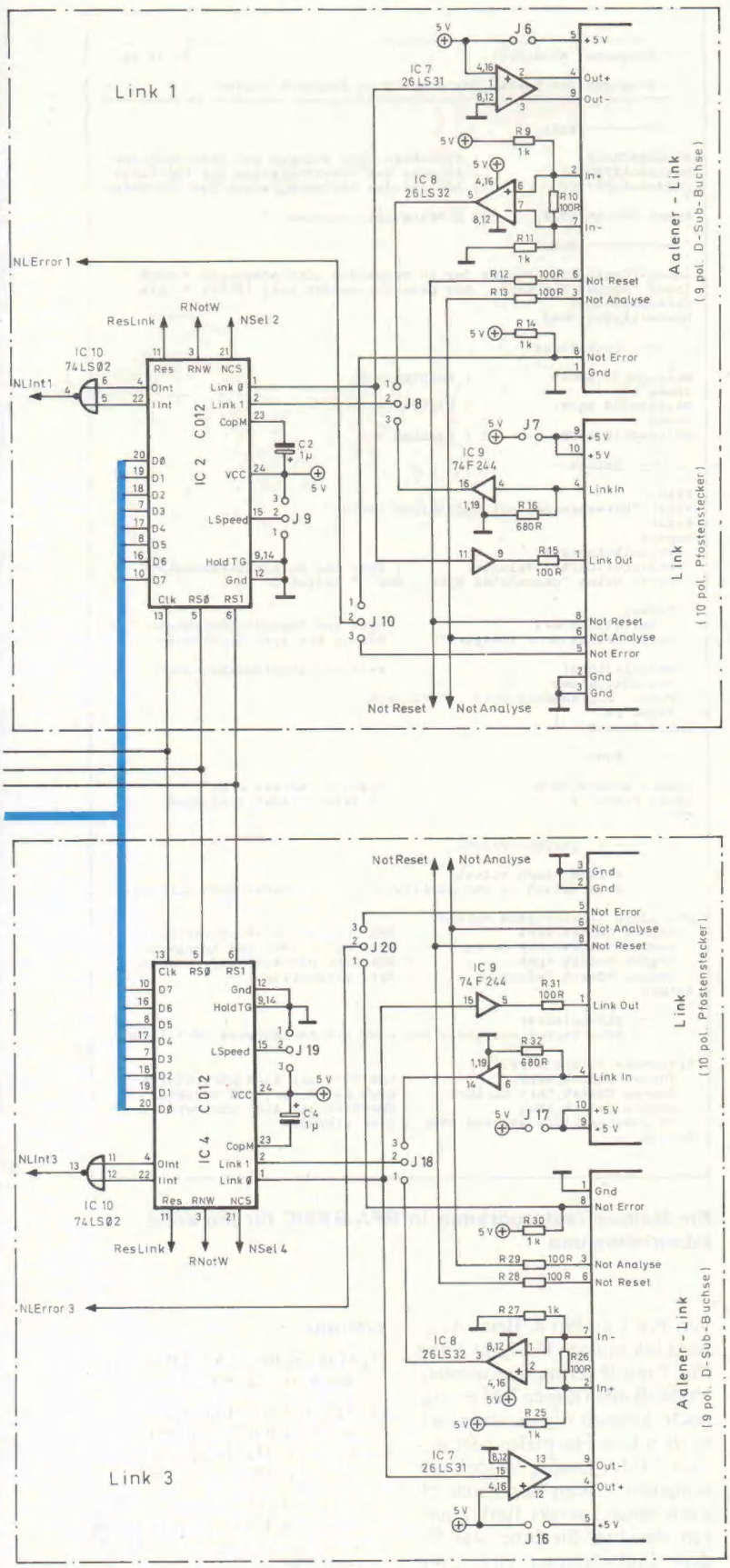
Der Aalener Link-Anschluß steht jeweils auch auf einem 10poligen Pfostenstecker zur Verfügung; es kann also wahlweise ein kurzes Flachbandkabel benutzt werden, das wiederum an eine 9polige Sub-D-Buchse angequetscht wird. So ist es möglich, die Buchsen nach Wunsch direkt auf der Platine oder am Gehäuse zu befestigen. Die Übertragungsrate läßt sich mit Jumpern für jedes Link getrennt einstellen (siehe Kasten). Jeder Link-Anschluß ist außerdem mit einem Jumper versehen, mit dem man die Stromversorgung zwischen Adapter und dem angeschlossenen Gerät trennen kann. Diese 5-V-Leitung dient üblicherweise zur Versorgung von kleineren Link-Adapter-Schaltungen (beispielsweise dem c't-Link-Centronics-Interface [4]). Bei getrennten Spannungsversorgungen kann diese Verbindung aber stören.



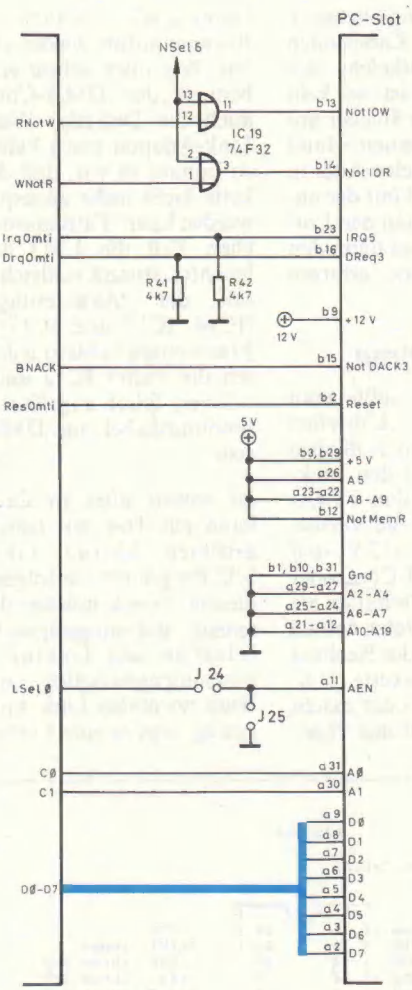
Interrupts

Die Interrupt-Ausgänge der Link-Bausteine werden beim Atari-Link-Adapter als Drq-Leitungen (Data Request) für die DMA-Übertragung mißbraucht. OInt (Output Interrupt) und IInt (Input Interrupt) werden jeweils durch ein NOR-Glied zusammengefaßt und auf einen 4-zu-1-Multiplexer in IC6 geschaltet. Der zweite Multiplexer ist mit den Fehler-Eingängen der Link-Anschlüsse verbunden. So kann jeweils über die Bits 3 und 4 des zweiten Latches (IC15) eine Link-Baugruppe für eine DMA-Übertragung oder den Interrupt-Betrieb scharf gemacht werden. Die Ausgänge des Multiplexers NLink und NError werden vom PAL weiterverarbeitet, welches sie gemäß der Einstellung von IntE0-2 (Interrupt Enable) auf NotInt und NotDrq des DMA-Ports weiterleitet. IntE0-2 sind mit den Bits 0 bis 2 des zweiten Latches (IC15) verbunden. Über diese Bits kann man verschiedene Konfigurationen einstellen.

Im Normalfall wird bei einer DMA-Übertragung die Konfiguration gewählt, bei der NLink mit NotDrq und StopDma mit NotInt verbunden sind. So wird bei Beendigung einer DMA-Übertragung ein Interrupt ausgelöst. Es läßt sich aber auch eine Konfiguration einstellen, bei der ein Interrupt beim Empfang des ersten Bytes über das Link erfolgt (siehe Kasten).



Die vier Link-Adapter sind identisch aufgebaut. Allerdings unterscheiden sich die Pin-Nummern, weshalb wir den Schaltplan in voller Breite abbilden.



Der Slot für einen OMTI-Controller erfordert nur zwei zusätzliche Gatter – ein nützliches Abfallprodukt.

Abfall-Slot

Für einen Slot, in dem man einen preisgünstigen PC-Festplattencontroller einstecken kann, fehlt nun nicht mehr viel. Die gesamte Adressierungs-Logik samt Puffer ist schon vorhanden. Tatsächlich besteht der zusätzliche Aufwand nur in der PC-Slot-Buchse und einem Gatter-Baustein, der die Signale IOW und IOR erzeugt. Die Belegung des Slots ist die gleiche wie in [2] beschrieben. In einem späteren Teil dieser Serie wird dann auch ein angepaßter Festplattentreiber vorgestellt werden.

Aufbau

IC7, 8 und 9 kann man jeweils wahlweise bestücken, je nachdem, ob man nur die normalen Links, die Aalener Link-Anschlüsse oder beide wechselweise benutzen will. Wer nur den PC-Slot zum Anschluß eines OMTI-Controllers benutzen möchte, braucht den Platinenteil mit den Link-Bausteinen überhaupt nicht. Der Anschluß an den DMA-Port des Rechners erfolgt über einen 19poligen Sub-D-Stecker. Da solche Stecker in gewinkelter Bauform zur Platinenmontage nicht leicht zu beschaffen sind, wurde auch ein 20poliger Pfostenstecker vorgesehen, an den ein Flachbandkabel per Quetsch-Verbinder angeschlossen werden kann.

Mit folgender Arbeitsmethode fällt es leicht, das Flachbandkabel auf der Gegenseite an den Stecker zu löten: Ritzen Sie mit einem scharfen Messer auf beiden Seiten flach die Isolierung aller Adern 3 mm vor dem Ende an. Dann ziehen Sie den Kunststoffmantel einfach jeweils von mehreren Adern gleichzeitig mit einer Flachzange ab. Nun können Sie mit einer flachen Schere die zusammenhängende Isolierung zwischen einzelnen Leitungen vorsichtig etwa 5 mm einschneiden und die Adern für die obere und untere Kontaktreihe des Steckers sortieren (dabei auf

richtige Verbindung achten). Verzinnen Sie die Kabelenden und auch die Lötkelche des Steckers. Danach ist es kein Problem mehr, den Stecker anzulöten. Mit der einen Hand drückt man jeweils eine Ader in einen Lötkelch, und mit der anderen Hand führt man den Lötkolben, mit dem man dann den Lötkelch nur kurz erhitzen muß.

Kontaktaufnahme

Nach dem Aufbau sollte man noch einmal alle Lötstellen sorgfältig überprüfen. Schließen Sie zum ersten Test den Link-Adapter allein an den DMA-Port an, und legen Sie Versorgungsspannung an (12 V sind nur für den OMTI-Controller nötig). Erst danach schalten Sie den Atari ST ein. Jetzt achten Sie bitte darauf, ob der Rechner korrekt von der Diskette 'bootet'. Passiert nichts oder macht die Floppy merkwürdige 'Step-Geräusche', so sollten Sie den Rechner sofort wieder abschalten. Wie oben schon erwähnt, bedient der DMA-Controller auch die Diskette. Weist der Link-Adapter einen Fehler auf, so kommt es vor, daß die Diskette nicht mehr angesprochen werden kann. Falls in einem solchen Fall die LED dauernd leuchtet, stimmt vielleicht etwas mit der 'Aktivierungslogik' (IC14, IC13 und IC17) nicht. Eine weitere Fehlerquelle könnten die Puffer IC12 und IC18 oder ein falsch angelötetes Verbindungskabel zum DMA-Port sein.

Ist soweit alles in Ordnung, kann ein Test mit dem abgedruckten kleinen GFA-BASIC-Programm erfolgen. Zu diesem Zweck müssen die Datenein- und -ausgangsleitungen (LinkOut und LinkIn) jeweils zusammengeschaltet werden. Beim normalen Link-Anschluß genügt jeweils eine Verbindung von Pin 1 an Pin 4. Beim Aalener Link müssen Pin 2 mit 4 und Pin 7 mit 9 verbunden werden. Verläuft auch dieser Test erfolgreich, können Sie die eventuell vorhandene Festplatte oder andere DMA-Geräte wieder anschließen. Prüfen Sie danach, ob auch diese korrekt funktionieren. Beachten Sie dabei, daß Sie dem Link-Adapter eine Controller-Nummer zuweisen müssen, die noch nicht belegt ist; SH204 oder SH205 sind übrigens werksseitig auf die Controller-Nummer 0 eingestellt.

(cp)

```
                              PAL20L8

Atari Link Adapter (ALIA)

                            _____ _____
                           |     \_/     |
Input          /BNRes -|  1        24  |-     Vcc
Input           IntE2 -|  2        23  |-   IntE1   Input
Input           IntE0 -|  3        22  |-    RWE   three out
Input             BRW -|  4        21  |-    RSel  three I/O
Input         DrqOmti -|  5        20  |-   DataE  three I/O
Input         StopDma -|  6        19  |- NDmaSel  three I/O
Input         IrqOmti -|  7        18  |- ResOmti  three I/O
Input         /NError -|  8        17  |-    NInt   three I/O
Input             Akt -|  9        16  |-     Drq   three I/O
Input             BA1 -| 10        15  |-     Int   three out
Input          /BNSel -| 11        14  |-  /NLink   Input
                  Gnd -| 12        13  |-  /BNAck   Input
                       |_____________|



c't/Helge Schulz/Carsten Meyer
Atari Link Adapter (ALIA)
Version 2.3
/BNRes  IntE2 IntE0 BRW DrqOmti StopDma IrqOmti /NError
Akt BA1 /BNSel Gnd /BNAck /NLink   Int Drq    Nint ResOmti
NDmaSel   DataE RSel RWE  IntE1 Vcc

/ResCmti =   /BNRes

/Int     =   NInt

/NInt    =   NError  * /IntE2 * /IntE1 *  IntE0
           + NLink   * /IntE2 *  IntE1 * /IntE0
           + IrqOmti * /IntE2 *  IntE1 *  IntE0
           + IrqOmti *  IntE2 * /IntE1
           + StopDma *  IntE2 *  IntE1

/Drq     =   /NLink * /DrqOmti
           + /IntE0 *  DrqOmti * /NLink
           + /IntE0 *  StopDma
           +  IntE0 * /DrqOmti *  NLink
           + /IntE2
           + /Akt

/DataE   =   BNSel
           + BNAck

/RWE     =   Akt * BRW * BA1

/RSel    =   BNSel * /BRW * /BA1

/NDmaSel =   Akt * BA1 * BNSel
           + Akt * BA1 * BNAck
```

Das PAL erzeugt alle wichtigen Signale zur Selektierung der verschiedenen Baugruppen und verwaltet die Interrupt-Quellen.

```
' ------------------------------------------------------------------
' - Programm: Alia-Test                                  27.10.88 -
' -                                                               -
' - Programm zum Testen des Atari Link Adapters (Alia)            -
' ------------------------------------------------------------------
'
' ---------- Init
'
Flock%=&H43E              ! Systemvar. zum Stoppen der Disk-Zugriffe
Ddata%=&HFF8604           ! Adresse des Datenregisters des DMA-Chips
Dmode%=&HFF8606           ! Adresse des Statusregisters des DMA-Chips
'
Spoke Flock%,&HFF         ! Diskzugriffe sperren
'
' ---------- Main
'
Input "Controller-Nummer des zu testenden Link-Adapters: ",Cnr%
Input "Nummer des Links, das getestet werden soll (0-3): ",Lnr%
Cnr%=&H20*Cnr%
Lnr%=(Lnr%+1)*&H4
'
' ----- Link-Reset
'
@Aliaout(0,&HC0)          ! ResLink = 0
Pause 1
@Aliaout(0,&HE0)          ! ResLink = 1
Pause 1
@Aliaout(0,&HC0)          ! ResLink = 0
'
' ----- Testen
'
Print
Print "Unterbrechen mit beliebiger Taste."
Print
Repeat
  Value1%=Rnd*255
  @Aliaout(Lnr%+1,Value1%)        ! Byte ins OutputDataRegister
  Print Using "gesendetes Byte : ###  ",Value1%;
  '
  Repeat
    @Aliain(Lnr%+2)               ! Byte aus InputStatusRegister
  Until Value%>0 Or Inkey$<>""    ! Warten bis Byte angekommen
  '
  @Aliain(Lnr%)                   ! Byte aus InputDataRegister
  Value2%=Value%
  Print "empfangenes Byte : ";Value2%
  Pause 10
Until Inkey$<>""
'
' ----- Ende
'
Sdpoke Dmode%,&H80                ! Diskette adressieren
Spoke Flock%,0                    ! Diskette wieder freigeben
End
'
' ---------- Unterprogramme
'
' ----- Aliaout(Adr%,Value%)
' ----- Byte Value% in ein Register mit der Adresse Adr% schreiben
'
Procedure Aliaout(Adr%,Value%)
  Sdpoke Dmode%,&H88              ! DMA-Port mit A1=0 adressieren
  Sdpoke Ddata%,Cnr% Or Adr%      ! Byte mit C.-Nr. und Adresse
  Sdpoke Dmode%,&H8A              ! DMA-Port mit A1=1 adressieren
  Sdpoke Ddata%,Value%            ! Byte einschreiben
Return
'
' ----- Aliain(Adr%)
' ----- Byte Value% aus einem Register mit der Adresse Adr% lesen
'
Procedure Aliain(Adr%)
  Sdpoke Dmode%,&H88              ! DMA-Port mit A1=0 adressieren
  Sdpoke Ddata%,Cnr% Or Adr%      ! Byte mit C.-Nr. und Adresse
  Sdpoke Dmode%,&H8A              ! DMA-Port mit A1=1 adressieren
  Value%=Dpeek(Ddata%) And &HFF   ! Byte einlesen
Return
```

Ein kleines Testprogramm in GFA-BASIC für die erste Inbetriebnahme

Literatur

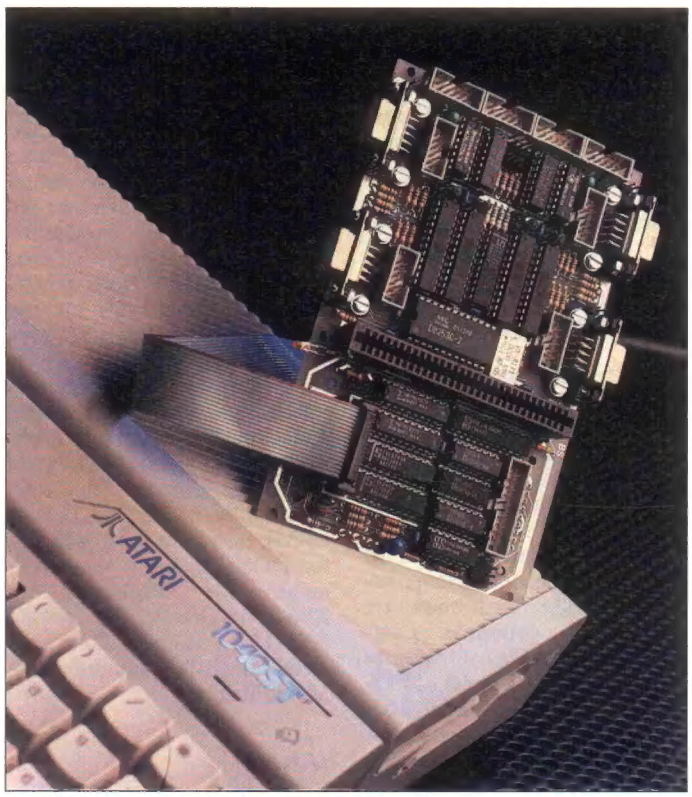
[1] Max Böhm, Der DMA-Port des Atari ST, c't 3/88

[2] Achim Freimann, Hans Werschner, Billiglösung, Preisgünstige PC-Harddisk am Atari ST, c't 2/88

[3] Manfred Helzle, Gar nicht link!, Die Link-Adapter-Chips C011 und C012 von INMOS, c't 11/87

[4] Manfred Helzle, Parallele Bindungen, Die TEK 4/8 bekommt eine universelle Parallel-Schnittstelle, c't 2/88

[5] Helfried Broer, Aalener Link-Connector, c't 12/88



EingekLinkt

Teil 2:
Programmierung des Atari-Link-Adapters

Helge Schulz

Nachdem nun die standesgemäße Hardware vorhanden ist, um den Atari ST in ein Transputer-System einzukLinken, dreht es sich in diesem Beitrag um die programmtechnische Seite des Atari-Link-Adapters (ALIA), dessen Schaltung im letzten Heft vorgestellt wurde. Was ist an Software nötig, um den ST als Hostrechner für eine TEK-4/8- oder KEK-4/8-Karte einzusetzen? Läßt sich das preisgünstige Occam-Programmiersystem OCS verwenden, das ja eigentlich nur für PC-kompatible Rechner gedacht war? Welche Klimmzüge muß der Programmierer machen, um eine DMA-Übertragung über Transputer-Links zu leiten?

Das OCS-System vom Transputer-Hersteller Inmos ist für PCs in Kombination mit einem Inmos-B004-Board [1] gedacht. Es besteht unter anderem aus Occam-II-Compiler, Linker und Dienstprogramm. Alle diese Programme laufen auf dem Transputer; für die B004-kompatible c't-TEK-4/8 ist deshalb keinerlei Anpassung nötig [2].

Aber auch der zum Dienstboten degradierte PC muß mit spezieller Software ausgestattet werden: damit ein Occam-Programm übersetzt werden kann, braucht man ein Programm, das den Compiler in den Transputer lädt, den Quelltext nach Bedarf übermittelt und anschließend den Code entgegennimmt und abspeichert. Diese Aufgaben übernimmt im ursprünglichen OCS-System das Programm AFSERVER.EXE (Alian File Server). Es stellt eine Reihe von Funktionen bereit, die über Nummern vom Transputer aufgerufen werden können. Darunter befinden sich zum Beispiel Funktionen zum Öffnen, Lesen, Schließen von Files und zur Ausgabe von Texten auf dem Bildschirm.

Daten-Butler

Soll das OCS-Paket auch auf anderen Systemen benutzt werden, braucht man einen solchen 'Daten-Butler' für das jeweilige Zielsystem – vorausgesetzt, ein Interface ist verfügbar. Die Anpassung wird dadurch vereinfacht, daß Inmos mit dem OCS-System den Quelltext des in C und Assembler geschriebenen AFSERVER ausliefert. Da bei den DOS-Aufrufen zwischen MSDOS und TOS keine großen Unterschiede bestehen, entstanden schon bald entsprechende Anpassungen für den Atari ST, bei denen man die Assembler-Routinen durch C-Routinen ersetzte. So etwa im März 1988 am Institut für theoretische Informatik der TU Braunschweig für einen Link-Adapter am ROM-Port.

Eine solche Anpassung hat allerdings den Nachteil, daß man sich damit auf das Protokoll des Inmos-Servers festlegt. Dieses Protokoll benutzt zum Beispiel zur Übertragung von Funktionsnummern, Fehlernummern und Flags grundsätzlich Langwörter (4 Bytes), obwohl nur ein einziges Byte notwendig wäre. Dadurch entsteht eine Menge (unnötiger) Kommunikations-'Overhead'. Der Ersatz der 8086-Assembler-Routinen in Form von C-Routinen, die nicht dieselbe Effizienz erreichen, bremst die Übertragung zusätzlich.

Im März experimentierte ich mit der ersten Version eines Link-Adapters für den DMA-Port, die zwar wunderbar funktionierte, aber den Anschluß weiterer DMA-Geräte blockierte. Der zweite Schaltungsentwurf mit durchgeschleiftem DMA-Bus funktionierte erst nach etlichen Versuchen in dieser Beziehung wirklich korrekt. Bei den vielen Änderungen an der Hardware hatte ich es schnell satt, dauernd meine Software neu anzupassen. So kam ich auf die Idee, zur Kommunikation über Links auf einer niedrigen Ebene eine Software-Schnittstelle zu bilden.

Die Überlegungen führten zu einem Assembler-Programm, mit dem das GEMDOS um Funktionen zum Senden und Empfangen von Bytes, Langwörtern und Blöcken über ein Link erweitert wird. Komplexere Programme wie etwa der OCS-Server oder ein Grafik-Server können dann, nachdem die GEMDOS-Erweiterung resident installiert ist, auf diese 'Low-Level-Routinen' zurückgreifen. Wird ein anderer Link-Adapter benutzt, so muß nur dieser 'Link-Treiber' gewechselt werden. Dieses Konzept hat sich bei mir bis heute gut bewährt.

Ein weiterer Vorteil besteht darin, daß der Treiber von jeder Programmiersprache aus angesprochen werden kann, die den Zugriff auf GEMDOS-Funktionen gestattet; die Routinen müssen nicht extra eingebunden werden. Selbst Interpreter-Sprachen, wie beispielsweise das populäre GFA-BASIC (siehe Beispiel), können so ohne Umstände auf den Link-Adapter zugreifen.

Individuell bedient

So kann der Anwender für jedes seiner Transputer-Programme den OCS-Server benutzen oder – wenn er es für günstig hält – ein spezielles Bedienungsprogramm in der Programmiersprache seiner Wahl schreiben. Im zweiten Fall kann das zugehörige Transputer-Programm natürlich nicht mehr auf andere Systeme übertragen werden, auf denen lediglich OCS-Server implementiert ist. Es ist sozusagen OCS-Server-inkompatibel geworden.

Die Vorstellung, der OCS-Server könne zu einem Software-Standard für die Bedienung von Transputern werden, war ohnehin illusorisch. Dazu ist der Server viel zu PC-spezifisch. Bei einem Datentransfer aus dem Speicher des Rechners zum Transputer beispielsweise muß die Adresse explizit in 8086-typischer Form als Segment und Offset (zwei 16-Bit-Worte) angegeben werden. Mit oder ohne OCS-Server – ein solches Transputer-Programm könnte sowieso nicht mit einem Hostrechner anderen Typs zusammenarbeiten.

Auch größere Software-Pakete wie etwa das Parallel C von Lattice Logic halten sich nicht an

die Vorgaben des OCS-Servers. Zwar übernimmt Parallel C weitgehend das Protokoll des Servers, aber er wurde um einige Funktionen erweitert, die der Compiler aufruft. Darunter ist zum Beispiel eine Funktion, die den Inhalt einiger Variablen des Servers zurückgibt. Die Funktion dieser Variablen ist aber nirgendwo dokumentiert!

Den Einsatz eines Treibers mit 'Low-Level-Routinen' halte ich auch deshalb für sinnvoll, weil es leichter sein dürfte, sich auf dieser Ebene auf einen Standard zu einigen als auf einer höheren. Ich will nicht behaupten, daß meine Lösung mittels einer GEMDOS-Erweiterung ideal sei. Sie soll aber eine Anregung sein. Hier bietet sich noch ein großes Betätigungsfeld für die DOIT (Deutsche Occam-Interessengemeinschaft der Transputer-Anwender). Vielleicht gelingt es ja irgendwann auch einmal, gerade in Hinblick auf zukünftige Transputer-Betriebssysteme, einen Kommunikationsstandard auf höherer Ebene festzulegen.

Das Treiber-Programm für den Atari-Link-Adapter habe ich 'ALIABIOS' getauft. Es stellt im wesentlichen die gleichen Funktionen zur Verfügung wie das Treiber-Programm für die TEK 4/8 am c't-Universalinterface [3], wurde aber um vier Funktionen zur Ausnutzung der DMA-Fähigkeit erweitert.

DMA-Übertragung

DMA steht für Direct Memory Access, zu deutsch direkter Speicherzugriff. Darunter versteht man eine Betriebsart, bei der ohne Mitwirkung des Mikroprozessors Daten im Speicher oder zwischen Peripherie und Speicher eines Computers transferiert werden. Den Transfer steuert ein spezieller Baustein; dieser sogenannte DMA-Controller wird mit der Startadresse, der Zieladresse und der Blockgröße programmiert und führt dann die Übertragung selbständig und recht schnell durch. Beim Atari ST läuft dies sogar im Hintergrund ab, das heißt, die CPU arbeitet gleichzeitig weiter. Es wird eine Übertragungsrate von bis zu 1,6 MByte pro Sekunde erreicht.

Die beiden Funktionen zum Senden und Empfangen im DMA-Modus starten also nur die Übertragung. Läuft sie erst einmal, kann das aufrufende Programm andere sinnvolle Dinge erledigen. Von Zeit zu Zeit sollte dieses Programm dann prüfen, ob der Transfer abgeschlossen ist. Wenn nicht, kann es die Übertragung auch vorzeitig beenden oder weiter warten.

Bitte nicht stören

Zu beachten ist, daß während einer DMA-Übertragung kein anderer Zugriff auf den DMA-Port erfolgen darf – weder durch eine andere Funktion des Treibers noch durch Aktionen im Zusammenhang mit Floppy Disk, Harddisk oder ein anderes Gerät am DMA-Port. Auch müssen die regelmäßigen Zugriffe der Interrupt-Routine unterbunden werden, die einen Diskettenwechsel registriert. Dies ist übrigens bei allen Programmen notwendig, die den DMA-Port nutzen. Abschalten kann man die Interrupt-Routine, indem man eine Systemvariable an der Adresse $43E mit einen Wert über 1 versieht; vor dem nächsten Diskettenzugriff muß sie wieder auf 0 gesetzt werden.

Die Beschreibung der zusätzlichen GEMDOS-Funktionen finden Sie in einem Extra-Kasten. Der Quelltext des Treiber-Programms kann leider nicht abgedruckt werden. Es hat einen erheblichen Umfang und würde zu viele Seiten füllen. Auch dürfte das Listing kaum dazu beitragen, die Programmierung des Link-Adapters zu erhellen. Der Treiber ist nämlich 'auf Tempo getrimmt'; wo immer möglich, werden benötigte Werte in den CPU-Registern gehalten. Im Lieferumfang der Version 2.0 des OCS-Servers für den Atari ST, erhältlich bei eMedia, Hannover, ist der Quelltext aber enthalten.

Im Detail

Um die Programmierung des Adapters zu erläutern, habe ich ein kleines Programm in GFA-BASIC geschrieben. Es soll exemplarisch die Programmschritte demonstrieren, die nötig sind, um eine DMA-Übertragung zu starten.

Nebenbei sieht man auch, wie Lese- und Schreiboperationen ohne DMA ablaufen. Diese finden sich in den Prozeduren AliaOut und AliaIn, die jeweils aus vier Anweisungen bestehen. Die ersten drei POKE-Anweisungen sind für die Adressierung erforderlich. Erst das letzte POKE beziehungsweise PEEK stellt den eigentlichen Lese- oder Schreibbefehl dar.

Das erste POKE in das Modus-Register des DMA-Controllers programmiert diesen so, daß der folgende Datenregister-Zugriff auf den DMA-Port geleitet wird, wobei die DMA-Portleitung A1 auf Low-Pegel geht. Die POKE-Anweisung in der nächsten Zeile überträgt somit das Byte mit der Controller-Nummer und der adapterinternen Adresse in das Latch IC 14 des Adapters. Ein weiteres POKE in das Modus-Register ist nötig, um A1 am DMA-Port wieder auf logisch 1 zu setzen. Erst jetzt kann mit einer PEEK- oder POKE-Anweisung der Lese- oder Schreibzugriff auf ein Register des Adapters erfolgen. Eine Liste der adapter-internen Adressen und der Register-Funktionen haben wir zusammen mit der Hardware-Beschreibung im letzten Heft abgedruckt.

Für eine DMA-Übertragung muß einiges mehr getan werden. Zunächst ist dem Adapter beziehungsweise dem DMA-Controller bekanntzugeben, wie lang der zu übertragende Block ist und an welcher Speicheradresse dieser sich befindet; dies geschieht in der Prozedur WriteAdr.

Wie schon im Zusammenhang mit der Hardware-Beschreibung erwähnt, kann der DMA-Controller eine Übertragung wegen eines internen Fehlers nicht richtig beenden. Dieses Manko wäre unter anderem beim Anschluß eines Transputers sehr nachteilig. Aus diesem Grund befindet sich auf dem Adapter ein Zählerbaustein (8253), der bei einem DMA-Transfer die Anzahl der übertragenen Bytes mitzählt. Der Zähler muß natürlich vor jeder Übertragung neu programmiert werden. Dies ist leider wegen einiger unangenehmer Eigenschaften des Chips nicht so einfach.

Der Baustein 8253 ist eigentlich zum Betrieb mit regelmäßigen Taktsignalen gebaut. Dabei fällt es nicht weiter auf, daß der Chip zu Beginn eines Zählvorgangs einen zusätzlichen Takt braucht, um die programmierten Ausgangswerte in sein Zählregister zu übernehmen. Will man den 8253 aber zum präzisen Abzählen einer Anzahl von Impulsen verwenden, macht sich diese Eigenart natürlich bemerkbar, weil das erste Ereignis 'übersehen' wird. Der Zählerausgang reagiert erst nach frühestens zwei Takten. Aus diesem Grund lassen sich die drei 16-Bit-Zähler, die der Baustein enthält, auch nicht kaskadieren. Auf dem Adapter sind zwei Zähler so zusammengeschaltet, daß sich beide Zählerwerte addieren. Dazu ist der Ausgang des ersten Zählers mit dem Freigabeeingang des zweiten verbunden, so daß dieser erst zu zählen beginnt, wenn der erste den programmierten Zählwert erreicht hat. Damit lassen sich dann maximal $2^{16} + 2^{16} = 1\,311\,072$ Bytes abzählen. Das

```
Name der Funktion (Parameterliste)
  Nummer (Stack-Korrektur-Wert)

SetLinkAdr(Adr)
  190 (4)

                    Bit 76543210
  Adr.w: Link-Adresse

         Bit 76543210
             000CCCLL

         LL  - Adapter-interne Link-Adresse (0 bis 3)
         CCC - DMA-Geräte-Adresse (0 bis 7)

         -1  Aktuelle Adresse abfragen

  Adressiert ein bestimmtes Link für alle weitere Aufrufe. Ist
  kein Adapter mit der angegebenen DMA-Geräte-Adresse vorhanden,
  wird in D0 -1 zurückgegeben. Wird -1 übergeben, wird in D0 die
  aktuelle Adresse zurückgegeben.

ByteToLink(Value)
  191 (4)

  Value.w: zu sendendes Byte

  Sendet ein Byte über das Link

ByteFromLink()
  192 (2)

  Nach dem Aufruf befindet sich in D0 das empfangene Byte.
```

LongWordToLink(Value)
103 (6)

Value.l: zu sendendes Langwort (32 Bit)

Sendet ein Langwort über das Link

LongWordFromLink(Ptr)
104 (6)

Ptr.l: Adresse im Speicher, an der das empfangene Langwort abgelegt werden soll

Empfängt ein Langwort über das Link

SliceToLink(Ptr,Len)
105 (10)

Ptr.l: Adresse des Speicherblocks, der über das Link gesendet werden soll

Len.l: Länge des zu sendenden Blocks in Bytes

Sendet einen Speicherblock über das Link

SliceFromLink(Ptr,Len)
106 (10)

Ptr.l: Adresse, an der der Datenblock abgelegt werden soll

Len.l: Länge des Datenblocks

Empfängt einen Datenblock über das Link

SToLinkDma(Ptr,Len)
107 (10)

Ptr.l: Adresse des Datenblocks, der über das Link gesendet werden soll

Len.l: Länge des Datenblocks in Bytes (4<Len<130561)

Sendet einen Speicherblock über das Link im DMA-Modus. Die Funktion wartet nicht, bis die Übertragung beendet ist, sondern kehrt nach dem Start der Übertragung sofort zurück.

SFromLinkDma(Ptr,Len)
108 (10)

Ptr.l: Adresse, an der der Datenblock abgelegt werden soll

Len.l: Länge des Datenblocks (4<Len<130561)

Empfängt einen Datenblock über das Link im DMA-Modus. Die Funktion wartet nicht, bis die Übertragung beendet ist, sondern kehrt nach dem Start der Übertragung sofort zurück.

StopDma()
109 (2)

Mit dieser Funktion kann eine DMA-Übertragung (auch vorzeitig) gestopt werden, wenn z. B. eine bestimmte Zeitspanne überschritten ist.

TestDma()
110 (2)

Prüft, ob eine DMA-Übertragung noch läuft. Wenn ja, wird in D0 eine 1 zurückgegeben, sonst 0.

TestError()
111 (2)

Testet die Error-Leitung des Transputers. Ist sie aktiv, wird in D0 eine 1 zurückgegeben, sonst eine 0. Mit Hilfe dieser Funktion kann auch geprüft werden, ob die GEMDOS-Erweiterung eingebunden ist. Wird weder 1 noch 0 zurückgegeben, sondern -32, ist sie nicht eingebunden.

SetReset()
112 (2)

Schickt über die Steuerleitungen ein Reset-Signal.

SetAnalyse()
113 (2)

Schickt über die Steuerleitungen ein Analyse-Signal.

BootRoot(Ptr)
114 (6)

Ptr.l: Adresse des Namens der Datei, die in den Transputer geladen werden soll (abgeschlossen mit einem 0-Byte)

Sendet ein File im DMA-Betrieb über das Link. Ist ein Transputer an das Link angeschlossen und schickt man ein Reset-Signal voraus, so kann man mit dieser Funktion sehr einfach ein Programm in den Transputer laden und starten.

Alle Funktionen, die senden und empfangen, haben ein "Timeout". Wird innerhalb einer gewissen Zeitspanne die Kommunikation nicht abgewickelt, so bricht die Funktion ab und gibt in D0 -1 zurück. Bei den Blockoperationen wird die Anzahl der tatsächlich ausgetauschten Bytes zurückgegeben, nicht aber bei einer DMA-Übertragung.

Diese zusätzlichen Funktionen stellt die GEMDOS-Erweiterung zur Verfügung. Sie ermöglichen einen einfachen Zugriff auf den Link-Adapter in den verschiedensten Programmiersprachen.

genügt, denn der DMA-Controller kann mit seinem eigenen (nur halb funktionierenden) 8-Bit-Blockzähler höchstens 255 Blöcke von jeweils 512 Bytes = 130560 Bytes 'in einem Rutsch' übertragen. Es sind wegen eines internen Fehlers nicht 256 × 512 Bytes, weil im 256. Block die letzten 16 Bytes aus dem internen FIFO-Puffer nicht mehr gesendet werden. Sollen größere Blöcke übertragen werden, muß der Block auf zwei kleinere aufgeteilt werden.

Um den 8253 zu programmieren, wird die gewünschte Byte-Anzahl im Programm in zwei Werte aufgeteilt. Die zwei zusätzlichen Impulse, die die beiden Zähler brauchen, müssen außerdem abgezogen werden. Nach Einstellung der Betriebsart der Zähler werden die beiden Werte in die Zähler geschrieben.

Die Blockadresse wird dem DMA-Controller in drei Schritten byteweise übermittelt. Dabei muß immer zuerst das niedrigstwertige Byte geschrieben werden, weil sonst die DMA-Übertragung aus schleierhaften Gründen nicht funktioniert. Die eigentlichen DMA-Routinen BlockOut und BlockIn machen zunächst die Interrupt-Ausgänge für die Übertragung scharf, weil sie ja als Handshake-Signale für den DMA-Controller mißbraucht werden. Mit dem zweiten Schritt schaltet das Programm diese Signale durch und legt den Ausgang des Zählers auf den Interrupt-Eingang des DMA-Ports.

Nun muß noch der Zähler des DMA-Controllers programmiert werden. Da jedoch der Zähler auf dem Adapter die eigentliche Aufgabe übernimmt, wird das Sector-Count-Register einfach auf seinen höchsten Wert (&HFF) eingestellt. Warum das Register wie im Programm zu sehen ist zu adressieren ist, wissen die Entwickler des DMA-Controllers.

Anschließend wird dasjenige Register des Adapters adressiert, über das die DMA-Übertragung laufen soll. In diesem Fall handelt es sich um ein Datenregister eines Link-Adapters. Mit der POKE-Anweisung, die den Wert &H0

```
' ------------------------------------------------------------------
' - Programm: Alia-DMA-Test                              28.11.88 -
' -                                                               -
' - Dieses Programm zeigt exemplarisch wie man eine DMA-          -
' - Übertragung mit dem Atari Linkadapter durchführt. Sinnvoll    -
' - ablaufen kann es nur, wenn auch ein entsprechender            -
' - Empfänger (Transputer, zweiter Atari) an einem Link ange-     -
' - schlossen ist.                                                -
' ------------------------------------------------------------------
'
' ---------- Init
'
Flock%=&H43E            ! Adresse zum Stoppen der Disk-Zugriffe
Ddata%=&HFF8604         ! Adresse des Datenregisters des DMA-Chips
Dmode%=&HFF8606         ! Adresse des Statusregisters "   "    "
Dbasel%=&HFF860D        ! Adressen für DMA-Adresse: Low-Byte
Dbasem%=&HFF860B        !                           Mid-Byte
Dbaseh%=&HFF8609        !                           High-Byte
Mfpio%=&HFFFA01         ! Ein-/Ausgaberegister des MC68901
'
Spoke Flock%,&HFF       ! Diskzugriffe sperren
'
' ---------- Main
'
Input "Controller-Nummer des zu testenden Link Adapters: ",Cnr%
Input "Nummer des Links der getestet werden soll (0-3) : ",Lnr%
Cnr%=&H20*Cnr%
Ladr%=(Lnr%+1)*&H4
'
' ----- Link-Reset
'
@Aliaout(&H0,&HC0)              ! ResLink = 0
Pause 1
@Aliaout(&H0,&H20)              ! ResLink = 1
Pause 1
@Aliaout(&H0,&HC0)              ! ResLink = 0
'
' ----- Testen
'
Repeat
  Print
  Input "Blockadresse: ",Adr%
  Input "Blocklänge  : ",Len%
  @Writeadr(Adr%,Len%)
  Print
  Print "E)mpfangen, S)enden oder A)bbruch ? ";
  Repeat
    In$=Upper$(Inkey$)
  Until In$="E" Or In$="S" Or In$="A"
  Print
```

```
  If In$="E" Or In$="S" Then
    If In$="S" Then
      @Blockout(Adr%,Len%)
    Else
      @Blockin(Adr%,Len%)
    Endif
  Endif
Until In$="A"
'
' ----- Ende
'
Sdpoke Dmode%,&H80                ! Diskette adressieren
Spoke Flock%,0                    ! Diskette wieder freigeben
End
'
' ---------- Unterprogramme
'
' ----- Blockout(Adr%,Len%)
' ----- Block der Länge Len% an der Adresse Adr% im DMA-Modus
' ----- senden
'
Procedure Blockout(Adr%,Len%)
  Print
  Print "Sende Block"
  '
  ' --- OutputInt aktivieren und auf DMA-Bus legen
  '
  @Aliaout(Ladr%+3,&H2)           ! Im C012 einschalten
  @Aliaout(&H0,&HC6 Or (Lnr%*8)) ! Beim Adapter durchschalten
  '                               ! (IntE0-2=110)
  '
  ' --- DMA-Controller programmieren
  '
  Sdpoke Dmode%,&H98              ! Sektor-Count-Register
  Sdpoke Dmode%,&H198             ! adressieren
  Sdpoke Ddata%,&HFF              ! Sektor-Count-Register = 256
  Sdpoke Dmode%,&H188             ! OutputDataRegister adressieren
  Sdpoke Ddata%,Cnr% Or Ladr%+1
  Sdpoke Dmode%,&H100             ! DMA-Übertragung starten
  '
  ' --- Auf Interrupt durch den Zähler warten
  '
  Repeat
  Until (Peek(Mfpio%) And &H20)=0
  '
  ' --- OutputInt wieder abschalten
  '
  @Aliaout(Ladr%+3,&H0)
  @Aliaout(&H0,&HC0)
  Print
  Print "Übertragung beendet"
Return
'
' ----- Blockin(Adr%,Len%)
' ----- Empfangenen Block der Länge Len% an der Adresse Adr%
' ----- ablegen
'
Procedure Blockin(Adr%,Len%)
  Print
  Print "Empfange Block"
  '
  ' --- InputInt aktivieren und auf DMA-Bus legen
  '
  @Aliaout(Ladr%+2,&H2)           ! Im C012 einschalten
  @Aliaout(&H0,&HC6 Or (Lnr%*8)) ! Im Adapter durchschalten
  '                               ! (IntE0-2=110)
  '
  ' --- DMA-Controller programmieren
  '
  Sdpoke Dmode%,&H98              ! Sektor-Count-Register
  Sdpoke Dmode%,&H198             ! adressieren
  Sdpoke Dmode%,&H98
  Sdpoke Ddata%,&HFF              ! Sektor-Count-Register = 256
  Sdpoke Dmode%,&H88              ! InputDataRegister adressieren
  Sdpoke Ddata%,Cnr% Or Ladr%     ! DMA-Übertragung starten
  Sdpoke Dmode%,&H0
  '
  ' --- Auf Interrupt durch den Zähler warten
  '
  Repeat
  Until (Peek(Mfpio%) And &H20)=0
  '
  ' --- InputInt wieder abschalten
  '
  @Aliaout(Ladr%+2,&H0)
  @Aliaout(&H0,&HC0)
  Print
  Print "Übertragung beendet"
Return
'
' ----- Writeadr(Adr%,Count%)
' ----- Anzahl Count% in den 8253 Zählerbaustein schreiben
' ----- Adresse Adr% in den DMA-Chip schreiben
'
Procedure Writeadr(Adr%,Count%)
  If Count%<4 Then
    Print
    Print "Weniger als 4 Bytes können nicht im DMA-Betrieb ";
    Print "übertragen werden !"
    End
  Endif
  '
  ' --- 8253 programmieren
  '
  If Count%>&H10004 Then                ! Werte für die beider
    Count1%=Count%-&HFFFF               ! Zähler errechnen
    Count2%=Count%-Count1%-2
  Else
    Count1%=Count%-4
    Count2%=2
  Endif
  Gosub Aliaout(&H17,&HB0)              ! Z2 auf Modus 0 stellen
  Gosub Aliaout(&H17,&H70)              ! Z1 auf Modus 0 stellen
  Gosub Aliaout(&H16,Count1% Mod &H100) ! Z2 programmieren
  Gosub Aliaout(&H16,Count1% Div &H100)
  Gosub Aliaout(&H15,Count2% Mod &H100) ! Z1 programmieren
  Gosub Aliaout(&H15,Count2% Div &H100)
  '
  ' --- Adresse in den DMA-Controller schreiben
  '
  Spoke Dbasel%,Adr%                ! immer zuerst das Low-Byte !!!
  Spoke Dbasem%,Adr% Div &H100
  Spoke Dbaseh%,Adr% Div &H10000
  '
Return
'
' ----- Aliaout(Adr%,Value%)
' ----- Byte Value% in ein Register mit der Adresse Adr% schreiben
'
Procedure Aliaout(Adr%,Value%)
  Sdpoke Dmode%,&H88                ! DMA-Port mit A1=0 adressieren
  Sdpoke Ddata%,Cnr% Or Adr%        ! Nr. und Adresse einschreiben
  Sdpoke Dmode%,&H8A                ! DMA-Port mit A1=1 adressieren
  Sdpoke Ddata%,Value%              ! Byte einschreiben
Return
'
' ----- Aliain(Adr%)
' ----- Byte Value% aus einem Register mit der Adresse Adr% lesen
'
Procedure Aliain(Adr%)
  Sdpoke Dmode%,&H88                ! DMA-Port mit A1=0 adressieren
  Sdpoke Ddata%,Cnr% Or Adr%        ! Nr. und Adresse einschreiben
  Sdpoke Dmode%,&H8A                ! DMA-Port mit A1=1 adressieren
  Value%=Dpeek(Ddata%) And &HFF     ! Byte einlesen
Return
```

Dieses kleine Demo-Programm in GFA-BASIC zeigt, wie eine DMA-Übertragung über den Link-Adapter gestartet wird.

für 'Block empfangen' oder &H100 für 'Block senden' in das Modus-Register schreibt, startet das Programm die DMA-Übertragung.

Danach kann sich der Mikroprozessor erst einmal ausruhen und warten oder andere sinnvolle Dinge tun. Das Ende einer DMA-Übertragung zeigt der Zählerbaustein an, indem er den Interrupt-Eingang des DMA-Ports auf 0 legt. Diese Leitung korrespondiert mit Bit 5 des Ein-/Ausgabeports im 68901 (MFP) und kann sehr einfach abgefragt werden. Bei entsprechender Programmierung des 68901 könnte man auch unmittelbar eine Interrupt-Routine starten.

Ist die DMA-Übertragung beendet, schaltet das Programm die Interrupts wieder aus. Dies ist erforderlich, weil ein anschließender Zugriff auf die Diskette oder auf ein anderes DMA-Gerät sonst möglicherweise gestört würde. (cp)

Literatur

[1] Inmos, Stand-alone occam 2 compiler, Beta 2 documentation, 1987 Inmos

[2] Heinz Ebert, Occamierung auf Transputern, Teil 1: Die ersten Schritte in Occam 2 auf der TEK 4/8, c't 1/88, S. 138

[3] Helge Schulz, Transputari, TEK 4/8 mit Businterface am Atari ST, c't 8/88, S. 112


```
PROC INMOS.ENTRY.POINT ()

  #USE "..\LIB\REALT4.OBJ"
  #USE "..\LIB\REALPDT4.OBJ"
  #USE "..\LIB\INTT4.OBJ"

  VAL MaxX     IS            640:
  VAL MaxB     IS         MaxX/32:
  VAL MaxY     IS            384:
  VAL MaxIter IS              40:
  VAL RLR      IS   -2.1(REAL32):
  VAL RRR      IS    1.0(REAL32):
  VAL IOR      IS   1.15(REAL32):

  PROC Iterate(VAL REAL32 xc,yc,INT k)
    REAL32 x,x2,y,y2:
    SEQ
      x:=0.0(REAL32)
      x2:=0.0(REAL32)
      y:=0.0(REAL32)
      y2:=0.0(REAL32)
      k:=0
      WHILE (((x2+y2) <= 4.0(REAL32)) AND (k < MaxIter))
        SEQ
          y:=((x*y)*2.0(REAL32))+yc
          x:=(x2-y2)+xc
          x2:=x*x
          y2:=y*y
          k:=k+1
```

```
    :
  PROC Erzeugen(VAL REAL32 fl,yc,[]INT Vektor)
    REAL32 xc:
    INT i:
    VAL Bits IS [$80000000,$40000000,$20000000,$10000000,
                 $08000000,$04000000,$02000000,$01000000,
                 $00800000,$00400000,$00200000,$00100000,
                 $00080000,$00040000,$00020000,$00010000,
                 $00008000,$00004000,$00002000,$00001000,
                 $00000800,$00000400,$00000200,$00000100,
                 $00000000,$00400000,$00200000,$00100000,
                 $00080000,$00040000,$00020000,$00010000,
                 $80000000,$40000000,$20000000,$10000000,
                 $08000000,$04000000,$02000000,$01000000]:
    SEQ
      SEQ i=0 FOR MaxB
        Vektor[i]:=0
      xc:=RLR
      SEQ ix=0 FOR MaxX
        INT k:
        REAL32 x1,y2:
        SEQ
          y2:=yc*yc
          x1:=xc+1.0(REAL32)
          IF
            (xc > (-0.75(REAL32)))
              REAL32 r,s:
              SEQ
                r:=(xc*xc)+y2
                SQRTP(s,(r-(0.5(REAL32)*xc))+0.0625(REAL32))
                IF
                  ((16.0(REAL32)*(r*s)) > (((5.0(REAL32)*s)-
                                           (4.0(REAL32)*xc))+
                                           1.0(REAL32)))
                    Iterate(xc,yc,k)
                  TRUE
                    k:=MaxIter
            TRUE
              IF
                (((x1*x1)+y2) > 0.0625(REAL32))
                  Iterate(xc,yc,k)
                TRUE
                  k:=MaxIter
          IF
            ((k REM 2)=0)
              Vektor[ix/32]:=Bits[ix REM 32] \/ Vektor[ix/32]
            TRUE
              SKIP
          xc:=xc+fl
```

```
    :
  PROC DoIt()
    VAL Halbe  IS (MaxY-1)/2:
    VAL Halbe1 IS Halbe+1:
    CHAN OF ANY Aus:
    PLACE Aus AT 0:
    [MaxB]INT V1,V2:
    REAL32 flx,fly,yc:
    SEQ
      yc:=IOR
      flx:=(RRR-RLR)/(REAL32 TRUNC(MaxX-1))
      fly:=(IOR+IOR)/(REAL32 TRUNC(MaxY-1))
      Erzeugen(flx,yc,V1)
      SEQ iy=0 FOR Halbe1
        SEQ
          yc:=yc-fly
          PAR
            Aus ! [V1 FROM 0 FOR MaxB]
            Erzeugen(flx,yc,V2)
          IF
            (iy < Halbe)
              SEQ
                yc:=yc-fly
                PAR
                  Aus ! [V2 FROM 0 FOR MaxB]
                  Erzeugen(flx,yc,V1)
            TRUE
              SKIP
      Aus ! [V2 FROM 0 FOR MaxB]
  :

  DoIt()
:
```

Einige Änderungen in dem Apfelmännchen-Programm aus c't 3/88 ermöglichen es, das vom Transputer berechnete Bild per DMA direkt in das Video-RAM zu übertragen.

```
' ------------------------------------------------------------
' - Programm: APFEL                                 01.12.88 -
' -                                                          -
' - Demoprogramm für GEMDOS-Erweiterung.                     -
' ------------------------------------------------------------
'
' --- Laufbandcode laden
' --- So umständlich ist es in Basic Maschinencode unterzubringen
Restore Mcode
Code$=""
Read S$
While S$<>"<<"
  Code$=Code$+Chr$(Val("&H"+S$))
  Read S$
Wend
Mcode:
Data 20,6F,00,04,D1,FC,00,00,7D,00,70,0F,E1,E0,72,26
Data E5,E0,51,C9,FF,FC,51,C8,FF,F4,4E,75,<<
' --- Laufbandtext laden
Restore Laufband
Lauf$=""
Read S$
While S$<>"<<"
  Lauf$=Lauf$+S$
  Read S$
Wend
Laufband:
Data "Dieses Apfelmännchen wird zur Zeit auf einem Transputer"
Data " errechnet. Die Bilddaten werden Zeilenweise per Link an"
Data " den Atari gesendet. Der Atari-Link-Adapter (ALIA)"
Data " empfängt sie dort und gibt sie an den DMA-Controller"
Data " weiter. Der schreibt die Daten direkt in das Video-RAM"
Data " des Ataris ! Die 68000 CPU hat im Moment nichts anderes"
Data " zu tun als die Uhr und das Laufband anzuzeigen !!!"
Data "  +++  "
Data "<<"
' --- Apfelmännchen-Programm in den Transputer laden
Cls
If Gemdos(111)<0 Then              ! TestError
  Alert$="Keine GEMDOS-Erweiterung|       geladen !"
  Alert 1,Alert$,1," Ende ",Void%
  End
Endif
Print
Input "DMA-Controller-Nummer (0-7): ",Cnr%
Input "Link-Nummer (0-3)          : ",Lnr%
If Gemdos(100,Cnr%*4+Lnr%)<0 Then       ! SetLinkAdr
  Alert$="    Kein Linkadapter mit|dieser Adresse angeschlossen !"
  Alert 1,Alert$,1," Ende ",Void%
  End
Endif
Cls
Print At(27,2);"Datei mit Transputerprogramm:"
Fileselect "*.B4","",N$
If N$="" Then
  End
Else
  N$=N$+Chr$(0)                    ! 0 Byte als Endzeichen anhängen
Endif
Cls
Void Gemdos(112)                   ! SetReset (Reset für Transputer)
Void Gemdos(114,L:Varptr(N$))      ! BootRoot (Programm laden)
' --- Bild zeichnen
Cls
Hidem
Llen%=Len(Lauf$)+1
Lindex%=1
Count%=0
Zeit%=Timer
Zeit$=Chr$(27)+Chr$(72)+"00:00:00"
Zeitptr%=Varptr(Zeit$)
Vptr%=Lpeek(&H44E)                 ! Adresse des Video-RAM -> Vptr%
Void Gemdos(108,L:Vptr%,L:30720)!  SFromLinkDma (DMA starten)
Running!=True
Repeat
  Cptr%=Varptr(Code$)
  Vsync                            ! mit Bildaufbau synchronisieren
  Void C:Cptr%(L:Vptr%)            ! Laufband zwei Punkte nach rechts
  Void C:Cptr%(L:Vptr%)
  If Gemdos(110)>0 Then            ! TestDma (DMA schon fertig ?)
    Z%=Timer-Zeit%                 ! Zeit für Uhr errechenen
    Poke Zeitptr%+2,48+Z% Div 120000 ! So gehts am schnellsten
    Poke Zeitptr%+3,48+(Z% Mod 120000) Div 12000
    Poke Zeitptr%+5,48+(Z% Mod 12000) Div 2000
    Poke Zeitptr%+6,48+(Z% Mod 2000) Div 200
    Poke Zeitptr%+8,48+(Z% Mod 200) Div 20
    Poke Zeitptr%+9,48+(Z% Mod 20) Div 2
    Print Zeit$
  Endif
  Count%=(Count%+1) And 3          ! Zeichen in Laufband einreihen
  If Count%=0 Then
    Text 631,397,Mid$(Lauf$,Lindex%,1)
    Inc Lindex%
    If Lindex%=Llen% Then
      Lindex%=1
    Endif
  Endif
  If Inkey$<>"" Then
    If Gemdos(110)>0 Then          ! TestDma (DMA schon fertig ?)
      Alert$="Ätsch, es geht trotzdem|       weiter..."
      Alert 1,Alert$,1,"   Ok    | Ende ",Button%
      Running!=(Button%=1)
    Else
      Running!=False
    Endif
  Endif
Until Not Running!
Void Gemdos(109)                   ! StopDma (immer stoppen)
Showm
```

Das GFA-Programm startet die Übertragung des Apfelmännchen-Bildes und demonstriert die Benutzung der GEMDOS-Erweiterung.